On the generation of equational dynamic logics for weighted imperative programs

Leandro Gomes[1], Alexandre Madeira[2,1], Manisha Jain[2] , Luis S. Barbosa[1,3]

[1] HASLab INESC TEC - Univ. Minho, Portugal
[2] CIDMA - Univ. Aveiro, Portugal
[3] QuantaLab, INL, Braga Portugal

Abstract. Dynamic logic is a powerful framework for reasoning about imperative programs. This paper extends previous work [9] on the systematic generation of dynamic logics from the propositional to the equational case, to capture 'full-fledged' imperative programs. The generation process is parametric on a structure specifying a notion of 'weight' assigned to programs. The paper introduces also a notion of bisimilarity on models of the generated logics, which is shown to entail modal equivalence with respect to the latter.

1 Introduction

The development of dynamic logic [3] along the past twenty years went hand-in-hand with the evolution of its object, i.e. *the very notion of a program*. The result was the emergence of a plethora of dynamic logics tailored to specific programming paradigms. This ranges from the well-known classical case [2] to less conventional examples for which e.g. programs are compositions of actions in UML state machines [6] or event/actions regular expressions [4]. Other rephrasing of what should count for a program in each specific context, lead to different variants of dynamic logics: Examples include probabilistic [7], fuzzy, concurrent [10], quantum [1] and continuous [11] computations, and combinations thereof.

Reference [9] initiated a research agenda on the systematic development of propositional, multi-valued dynamic logics parametric on an algebraic structure, actually an action lattice, which defines both the computational paradigm where programs live, and the truth space where assertions take value. This paper extends this agenda to a new level, taking computational states as valuations of variables over a given domain, and programs as their modifiers. The idea is to capture typical imperative programs and their interpretation over different notions of 'weighted' computation — the very notion of *weight* being brought to scene as a parameter, encoded in the action lattice, for the generation of the corresponding dynamic logic. Depending on each action lattice chosen, such weights

⋆ This work was founded by the ERDF — European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project POCI-01-0145-FEDER-030947. The second author is supported in the scope of the framework contract foreseen in the numbers 4, 5 and 6 of the article 23, of the Decree-Law 57/2016, of August 29, changed by Portuguese Law 57/2017, of July 19.

will be interpreted as e.g. vagueness degree associated to the effectiveness of a particular computation, or a measure of the resources consumed in it, or even the associated cost or execution time.

Note that in all approaches discussed in the literature, even when some form of structured computation is considered, validity of assertions is always stated in classical terms. The approach proposed here goes a step further in the sense that validity of structured computation (e.g. fuzzy, costed, timed) is discussed in a logic capturing itself the corresponding notion of behaviour.

Differently from our previous work [9], 'fully-fledged' programs are considered here. This means that assignment of values from a data space to a variable is taken as the elementary construction, programs being defined over an equational signature of program variables, predicate and function symbols. Thus, in the sequel, programs are expressions generated by the following grammar:

$$\pi ::= x := t \mid \pi; \pi \mid \textbf{if } c \textbf{ then } \pi \textbf{ else } \pi \textbf{ fi} \mid \textbf{while } c \textbf{ do } \pi \textbf{ od} \tag{1}$$

where t denote terms with variables from a set X.

Bisimulation is defined parametrically on an action lattice, over the resulting computational models. Finally, bisimilarity is shown to entail modal equivalence for the corresponding dynamic logic.

The remaining of this paper is organised as follows. After a brief background overview in Section 2, to recap the definition of an action lattice and some of its fundamental properties, Section 3 extends the method proposed in [9] to incorporate 'fully-fledged' imperative programs, i.e. program variables and assignments. All constructions are illustrated in detail for three paradigmatic parameters: classical Boolean lattices, Gödel algebras to capture vagueness in computation, and the tropical semiring to reason about resource consumption. Bisimilarity and an invariance result is discussed, as a second contribution of the paper, in Section 4. Finally, Section 5 concludes, and enumerates topics for future work.

2 Action Lattices

As explained in the Introduction, the construction of multi-valued, equational, dynamic logics is parametric on an action lattice which induces both the computational model for programs and the truth space for logics. This section recalls the relevant definition and properties [9].

Definition 1. *An* action lattice *is a tuple*

$$\mathbf{A} = (A, +, ;, \mathbf{0}, \mathbf{1}, ^*, \rightarrow, \cdot)$$

where A is a set, $\mathbf{0}$ and $\mathbf{1}$ constants, and $+, ;, \rightarrow$ and $\cdot$ binary operations and * a unary operation in A satisfying the axioms in Figure 1, where the relation $\leq$ is induced by $+$: $a \leq b$ iff $a + b = b$.

$$a + (b + c) = (a + b) + c \quad (2) \qquad \mathbf{1} + a + (a^*; a^*) \leq a^* \quad (11)$$
$$a + b = b + a \quad (3) \qquad a; x \leq x \Rightarrow a^*; x \leq x \quad (12)$$
$$a + a = a \quad (4) \qquad x; a \leq x \Rightarrow x; a^* \leq x \quad (13)$$
$$a + \mathbf{0} = \mathbf{0} + a = a \quad (5) \qquad a; x \leq b \Leftrightarrow x \leq a \to b \quad (14)$$
$$a; (b; c) = (a; b); c \quad (6) \qquad a \cdot (b \cdot c) = (a \cdot b) \cdot c \quad (15)$$
$$a; \mathbf{1} = \mathbf{1}; a = a \quad (7) \qquad a \cdot b = b \cdot a \quad (16)$$
$$a; (b + c) = (a; b) + (a; c) \quad (8) \qquad a \cdot a = a \quad (17)$$
$$(a + b); c = (a; c) + (b; c) \quad (9) \qquad a + (a \cdot b) = a \quad (18)$$
$$a; \mathbf{0} = \mathbf{0}; a = \mathbf{0} \quad (10) \qquad a \cdot (a + b) = a \quad (19)$$

Fig. 1. A possible axiomatisation of action lattices.

An action lattice $\mathbf{A}$ is *complete* when every subset of its carrier A has both supremum and infimum with respect to $\leq$. The greatest and least elements are denoted in the sequel by $\top$ and $\bot$, respectively. Note that in any action lattice $\bot = 0$, since for any $a \in A$, $a + 0 = a$, i.e. $0 \leq a$. Consider a non-empty set I. We say that $\mathbf{A}$ is *linear* if it satisfies, for any set $\{a_i | i \in I\}$, the property

$$\sum_{i \in I} a_i = a_j, \text{ for some } j \in I \quad (20)$$

Since operators $+$, ; and $\cdot$ are associative, they admit a n-ary iterated version, represented by $\sum$, $\prod$ and $\bigwedge$, respectively. Note that the structure $(A, +, ;, \mathbf{0}, \mathbf{1}, ^*)$ axiomatised by (2) - (13) forms a Kleene algebra.The following handy properties are easily proved [9]:

$$x \leq y \Rightarrow x; a \leq y; a \quad (21)$$
$$a \leq b \;\&\; c \leq d \Rightarrow a + c \leq b + d \quad (22)$$

The generation of dynamic logics illustrated in the following sections will be parametric on the class of complete action lattices. Actually, completeness is required to guarantee the existence of infinite sums. The following are examples of complete action lattices, with which the proposed constructions will be illustrated along the paper.

Example 1. The first example is the Boolean lattice

$$\mathbf{2} = (\{\top, \bot\}, \vee, \wedge, \bot, \top, ^*, \to, \wedge)$$

with the standard interpretation of Boolean connectives. Operator * maps each element of $\{\top, \bot\}$ to $\top$, and $\to$ corresponds to logical implication.

Example 2. Gödel algebras are the locally finite variety of Heyting algebras. Formally,

$$\mathbf{G} = ([0, 1], max, min, 0_{\mathbf{G}}, 1, ^*, \to, min)$$

where

$$x \to y = \begin{cases} 1, & \text{if } x \leq y \\ y, & \text{if } y < x \end{cases}$$

Example 3. Finally, the $(min, +)$ Kleene algebra [8], known as the *tropical semiring*, can be extended to an action lattice through the introduction of residuation $\to$:

$$\mathbf{R} = (\mathbb{R}_0^+ \cup \{+\infty\}, min, +_{\mathbf{R}}, +\infty, 0_{\mathbf{R}}, ^*, \to, min)$$

where, for any $x, y \in \mathbb{R}_0^+ \cup \{+\infty\}$, $x^* = 0_{\mathbf{R}}$ and $x \to y = max\{y - x, 0\}$, with $\mathbb{R}_0^+ = \{x \in \mathbb{R} \mid x \geq 0\}$.

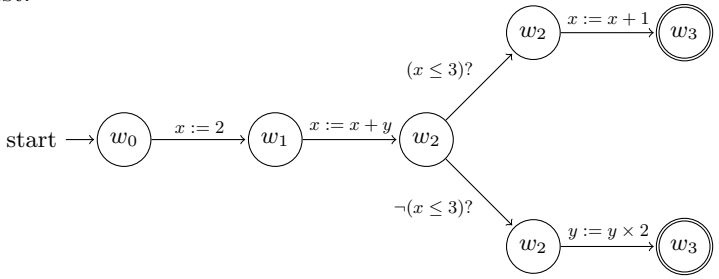
3 Generation of Equational, Dynamic Logics

Each complete action lattice $\mathbf{A}$ induces a multi-valued, equational dynamic logic $\Gamma(\mathbf{A})$ to reason, as explained above, about 'full-fledged' imperative programs with weighted computations interpreted over $\mathbf{A}$. Such programs are generated as indicated in (1).

Example 4. This toy program over a set of variables $\{x, y\}$ and the real numbers as data space will be used for illustration purposes in the sequel.

$$x := 2; x := x + y; (\textbf{if } x \leq 3 \textbf{ then } x := x + 1 \textbf{ else } y := y \times 2)$$

Note that its execution can be represented by the following transition system, where the conditional statement is encoded as a sum of alternatives guarded by a test.



Let us start by carefully fixing the syntactic support for the generated logics. Programs are defined over a data signature $\Sigma = (F, P)$, where F and P denote sets of function and predicate symbols, respectively. As usual, let notation $T_\Sigma(X)$ stand for the set of Σ-terms with variables in X, and represent by $T_\Sigma^F(X)$ (respectively, $T_\Sigma^P(X)$) its restriction to functional (respectively, predicate) terms. Thus,

$$\mathrm{Prg}_0(\Sigma, X) = \{x := t \mid x \in X \text{ and } t \in T_\Sigma(X)\}$$

defines the set of *atomic* programs for the pair (Σ, X), from which an arbitrary (composed) program is generated as an expression described by the following rule

$$\pi ::= \pi_0 \mid \phi? \mid \pi; \pi \mid \pi + \pi \mid \pi^*$$

with $\pi_0 \in \mathrm{Prg}_0(\Sigma, X)$, and $\phi?$ standing for a suitable notion of *test*. The latter, however, needs to be handled with some care: indeed the meaning of a test depends on the logic $\Gamma(\mathbf{A})$, and therefore on $\mathbf{A}$ itself, as we will discuss below on defining its semantics in terms of the satisfaction relation for $\Gamma(\mathbf{A})$. For the moment, it is enough to notice that choice $(+)$, iteration $(\,^*)$ and and tests $(\phi?)$ encode the usual 'syntactic sugar' constructs for conditionals and loops as considered in rule (1). The set of composed programs for (Σ, X) is denoted by $\mathrm{Prg}(\Sigma, X)$.

Once a language for programs is fixed, the set of formulas for $\Gamma(\mathbf{A})$ introduces, as expected, the universal and existential modalities over programs. Formally,

Definition 2. *A* signature *for $\Gamma(\mathbf{A})$ is a tuple*

$$\Delta = (\Sigma, \Pi)$$

where Σ is a data signature and $\Pi \subseteq \mathrm{Prg}_0(\Sigma, X)$ is a set of variable assignments. The set of formulas *for Δ, denoted by $\mathrm{Fm}^{\Gamma(\mathbf{A})}(\Delta)$, are the ones generated by the rule*

$$\varphi ::= \top \mid \bot \mid p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \to \varphi \mid \langle \pi \rangle \varphi \mid [\pi]\varphi$$

for $p \in T^P_\Sigma(X)$ and π is a program in $\mathrm{Prg}(\Sigma, X)$ that only uses atomic programs in Π.

Note that we sometimes make use of $\neg\varphi$ as an abbreviation for $\varphi \to \bot$, as in Example 4.

We can now turn to semantics. For each $\mathbf{A}$, models are defined over state spaces whose elements are graded valuations of variables, i.e. functions $w : X \times \mathbb{R} \to A$, where A is the carrier of action lattice $\mathbf{A}$. We denote the set of all states by $A^{X \times \mathbb{R}}$.

Definition 3 (Models). *Let $\Delta = (\Sigma, \Pi)$ be a signature and X a set of variables. A $\Gamma(\mathbf{A})$-model for Δ is a structure*

$$M = (W, E)$$

where

- *$W \subseteq A^{X \times \mathbb{R}}$ is a set of states;*
- *$E : \Pi \times (W \times W) \to A$ is a program grading function.*

The set of $\Gamma(\mathbf{A})$-models for Δ is denoted by $\mathrm{Mod}^{\Gamma(\mathbf{A})}(\Delta)$.

Intuitively the value of $E(\pi, (w_0, w_1))$ represents the graded execution of program π from state w_0 to w_1, i.e. the weight associated to corresponding transition. For instance, in Example 4, taking $\mathbf{A}$ as a Gödel algebra (Example 2), the expression $E(x := 2, (w_0, w_1)) = 0.6$ would mean that the system allows the execution of the assignment $x := 3$ from state w_0 to w_1 with 0.6 as a degree of certainty. Note that these values are attributed in the model. The interpretation of functional terms and predicates becomes as detailed in the following definitions.

Definition 4 (Interpretation of functional terms). *Let $\Delta = (\Sigma, \Pi)$ be a signature and $M \in \mathrm{Mod}^{\Gamma(\mathbf{A})}(\Delta)$. The interpretation of a functional term $t \in T^F_\Sigma(X)$ in M, for each $w \in W$, is given by the map*

$$[\![t]\!]_w : T^F_\Sigma(X) \to A^{\mathbb{R}}$$

defined recursively as follows:

- $[\![x]\!]_w(r) = w(x, r)$
- $[\![c]\!]_w(r) = \begin{cases} \mathbf{1} & \text{if } r = c \\ \mathbf{0} & \text{otherwise} \end{cases}$
- $[\![f(t_1, \ldots, t_n)]\!]_w(r) = \sum_{i \in I} \{\prod_{j=1}^{n} [\![t_j]\!]_w(r^i_j) \mid f(r^i_1, \ldots, r^i_n) = r\}$, *where I is the cardinality of the set of all possible solutions of $f(r^i_1, \ldots, r^i_n) = r$ in $\mathbb{R}$, with each f of arity n being interpreted as a function on real numbers $\mathbb{R}^n \to \mathbb{R}$ (e.g. $+$, $\times$, ^{2}, $\sqrt{}$, ...).*

where $x \in X$ and c is the syntactic representation of the constant $c \in \mathbb{R}$.

Example 4 may also help in illustrating this issue. Consider a model $M = (W, E)$, w_0, w_1, $w_2 \in W$, $X = \{x, y\}$, and the complete action lattice $G = ([0,1], max, min, 0, 1, ^*, \to, min)$ of Example 2. Take $[\![x]\!]_{w_0}(1) = w_0(x, 1) = 0.5$, $[\![x]\!]_{w_0}(2) = w_0(x, 2) = 0.2$, $[\![y]\!]_{w_0}(1) = w_0(y, 1) = 0.1$, $[\![y]\!]_{w_0}(2) = w_0(y, 2) = 0.4$ and 0 otherwise for state w_0. The interpretation of the term 2 in w_0 is given by $[\![2]\!]_{w_0}(2) = 1$ and 0 otherwise. The interpretation of the term $x + y$ in w_0 is given by:

$$\begin{aligned}
[\![x+y]\!]_{w_0}(2) =& [\![x]\!]_{w_0}(1); [\![y]\!]_{w_0}(1) = min\{0.5, 0.1\} = 0.1 \\
[\![x+y]\!]_{w_0}(3) =& [\![x]\!]_{w_0}(1); [\![y]\!]_{w_0}(2) + [\![x]\!]_{w_0}(2); [\![y]\!]_{w_0}(1) \\
=& w_0(x, 1); w_0(y, 2) + w_0(x, 2); w_0(y, 1) \\
=& max\{min\{0.5; 0.4\}\}, min\{0.2; 0.1\}\} = 0.4 \\
[\![x+y]\!]_{w_0}(4) =& [\![x]\!]_{w_0}(2); [\![y]\!]_{w_0}(2) = min\{0.2, 0.4\} = 0.2
\end{aligned}$$

and 0 otherwise.

Definition 5 (Interpretation of predicates). *Let Δ be a signature and $M \in \mathrm{Mod}^{\Gamma(\mathbf{A})}(\Delta)$. The interpretations of a predicate $p \in T^P_\Sigma(X)$ in M is given by the map*

$$[\![p]\!]_w : T^P_\Sigma(X) \to A$$

defined by

$$[\![p(t_1, \ldots, t_n)]\!]_w = \sum_{i \in I} \{\prod_{j=1}^{n} [\![t_j]\!]_w(r^i_j) \mid p(r^i_1, \ldots, r^i_n) \text{ is true}\}$$

where I is the cardinality of the set of all possible values $(r^i_1, \ldots, r^i_n) \in \mathbb{R}^n$ satisfying $p(r^i_1, \ldots, r^i_n)$, with each p of arity n being interpreted as a function over terms $T^F_\Sigma(X)$ like boolean predicate symbols (e.g. $\leq$, $=$, ...).

Again this can be illustrated by computing the truth degree of predicate $x \leq 3$ in state w_2, of Example 4. $[\![x \leq 3]\!](w_2) = [\![x]\!]_{w_2}(3); [\![3]\!]_{w_2}(3)$:

G: $min\{0.3, 1\} = 0.3$. The value 0.3 means that the predicate is true with a certainty 0.3.

R: $1.2 +_{\mathbf{R}} 3.7 = 4.9$. This interpretation corresponds to the energy consumed by evaluating the predicate.

Definition 6 (Interpretation of atomic programs). *The interpretation of atomic programs in a $\Gamma(\mathbf{A})$-model $M \in \mathrm{Mod}^{\Gamma(\mathbf{A})}(\Delta)$ is a map*

$$[\![\_]\!]_0 \colon \Pi \to A^{W \times W}$$

mapping each $x := t \in \Pi$ into function

$$[\![x := t]\!]_0(w, w') = \begin{cases} E(x := t, (w, w')) & \text{if } (w, w') \in (\!|x := t|\!) \\ \boldsymbol{0} & \text{otherwise} \end{cases}$$

where $(\!|x := t|\!)$ is the standard relational semantics of a program assignment, typically given by:

$$(w, w') \in (\!|x := t|\!) \Leftrightarrow \begin{cases} w'(y, r) = w(y, r) & \text{if } y \neq x \\ w'(x, r) = [\![t]\!]_w(r) & \text{otherwise} \end{cases}$$

This is made concrete by interpretation in each of the three distinct models of computation considered in the paper, as captured in the action lattices of examples 1, 2 and 3, respectively.

2: The degree of certainty of execution is bivalente: either $\top$ or $\bot$, coinciding with the classical setting where an action simply may or may not execute.

G: Assume $[\![x := 2]\!]_0(w_0, w_1) = E(x := 2, (w_0, w_1)) = 0.8$, $[\![x := x + y]\!]_0(w_1, w_2) = E(x := x + y, (w_1, w_2)) = 0.4$, $[\![x := x + 1]\!]_0(w_2, w_3) = E(x := x + 1, (w_2, w_3)) = 0.7$ and $[\![y := y \times 2]\!]_0(w_2, w_3) = E(y := y \times 2, (w_2, w_3)) = 0.9$. Such values are regarded as degrees of certainty, or, in a complementary reading, vagueness, associated to the execution of actions $x := 2$, $x := x + y$, $x := x + 1$ and $y := y \times 2$, respectively.
As a consequence of executing these assignments, the weights of the variables are updated accordingly in the next state. That is the case of x in state w_1, by assuming the value $w_1(x, 1) = [\![2]\!]_{w_0}(2) = 1$, and 0 otherwise, according to definition 6. The weights of y are maintained, since the assignment $x := 2$ does not modify the value of y. The situation may be interpreted as follows: from a state where property $x = 1$ has a truth degree of 0.5 and $x = 2$ has a truth degree of 0.2, the execution of action $x := 2$ with a certainty value of 0.8, whenever occurs, leads to a state where $x = 2$ is true (i.e. has 1 as its truth degree). The weights of the variable x in w_2 are updated as follows:

$$w_2(x, 3) = [\![x + y]\!]_{w_1}(3) = [\![x]\!]_{w_1}(2); [\![y]\!]_{w_1}(1) = min\{1, 0.1\} = 0.1$$
$$w_2(x, 4) = [\![x + y]\!]_{w_1}(4) = [\![x]\!]_{w_1}(2); [\![y]\!]_{w_1}(2) = min\{1, 0.4\} = 0.4$$

R: Consider, for example, $E(x := 2, (w_0, w_1)) = 8$, $E(x := x+y, (w_1, w_2)) = 4$, $E(x := x + 1, (w_2, w_3)) = 7$ and $E(y := y \times 2, (w_2, w_3)) = 9$. These values can be regarded as resources (e.g. energy) consumed by executing the associated actions. Analogously to the previous case, the weights associated to y are kept.

Finally, to interpret an arbitrary program in $Prg(\Sigma, X)$ one proceeds in two steps. First, the semantics of composed program constructs is given directly in terms of operations on A-valued binary relations $A^{W \times W}$: union, composition, and Kleene closure. To interpret such operators, we define the following algebra:

Definition 7. *Let* $\mathbf{A} = (A, +, ;, 0, 1, *, \rightarrow, \cdot)$ *be an action lattice and* W *be a finite set of states. The algebra of* program grading functions *is the structure*

$$\mathbf{E} = (Z(E), \cup, \circ, \varnothing, \chi, *)$$

where:

- $Z(E)$ *is the universe of all the program grading functions*
- $(E(\pi_1) \cup E(\pi_2))(w, w') = E(\pi_1, (w, w')) + E(\pi_2, (w, w'))$
- $(E(\pi_1) \circ E(\pi_2))(w, w') = \sum_{w'' \in W} E(\pi_1, (w, w'')); E(\pi_2, (w'', w'))$
- $\varnothing(w, w') = \mathbf{0}$
- $\chi(w, w') = \begin{cases} \mathbf{1}, & \text{if } w = w' \\ \mathbf{0}, & \text{otherwise} \end{cases}$
- $(E(\pi))^*(w, w') = \sum_{i \geq 0} (E(\pi))^i(w, w') = (E(\pi))^0(w, w') + (E(\pi))^1(w, w') + (E(\pi))^2(w, w') + \ldots$

with $E(\pi_1), E(\pi_2) \in Z(E)$.

Note that operator $*$ can be defined as an infinite sum due to the completeness of the action lattice.

Definition 8. *Let* $M \in \mathrm{Mod}^{\Gamma(\mathbf{A})}(\Delta)$ *be a model of* $\Gamma(\mathbf{A})$. *The interpretation of a program* $\pi \in \mathrm{Prg}(\Sigma, X)$ *is a map*

$$[\![-]\!] \colon \mathrm{Prg}(\Sigma, X) \to A^{W \times W}$$

recursively defined by

- $[\![\pi_0]\!] = [\![\pi_0]\!]_0$, *for each* $\pi_0 \in \mathrm{Prg}_0(\Delta)$
- $[\![\pi; \pi]\!] = [\![\pi]\!] \circ [\![\pi']\!]$
- $[\![\pi + \pi]\!] = [\![\pi]\!] \cup [\![\pi']\!]$
- $[\![\pi^*]\!] = [\![\pi]\!]^*$.

where, for $r \in A^{W \times W}$, $r^*(w, w') = \sum_{k \geq 0} r^k(w, w')$.

Again Example 4 can be called to illustrate choice and sequential composition by interpreting fragments $(x := 2); (x := x + y)$ and $(x := x + 1) + (y := y \times 2)$. The first one yields,

$$\begin{aligned}
[\![x := 2; x := x + y]\!](w_0, w_2) &= ([\![x := 2]\!]_0 \circ [\![x := x + y]\!]_0)(w_0, w_2) \\
&= [\![x := 2]\!]_0(w_0, w_1); [\![x := x + y]\!]_0(w_1, w_2) \\
&= E(x := 2, (w_0, w_1)); E(x := x + y, (w_1, w_2))
\end{aligned}$$

which can be instantiated within the three usual lattices we have been considering:

2: Under this interpretation programs either fail or succeed. In the absence of failure execution proceeds sequentially; otherwise, if one (or both) fails (takes 'weight' $\perp$), so does the composite.

G: In this case a degree of confidence, or certainty, is associated to the composition based on the corresponding degree for the atomic components. This is computed as a minimum. For example, if $E(x := 2, (w_0, w_1)) = 0.8$ and $E(x := x + y, (w_1, w_2)) = 0.4$ the overall confidence degree for the composition becomes $min\{0.8, 0.4\} = 0.4$.

R: Computations have a cost, under this interpretation, for example the amount of energy dissipated. Thus, $E(x := 2, (w_0, w_1)); E(x := x+y, (w_1, w_2)) = 8 +_{\mathbf{R}} 4 = 12$ represents the sum of the energy consumed by both atomic programs $x := 2$ and $x := x + y$.

The interpretation of $(x := x + 1) + (y := y \times 2)$, on the other hand, is given by

$$\begin{aligned}
[\![(x := x + 1) + (y := y \times 2)]\!](w_2, w_3) = \ & ([\![x := x + 1]\!]_0 \cup [\![y := y \times 2]\!]_0)(w_2, w_3) \\
= \ & [\![x := x + 1]\!]_0(w_2, w_3) + [\![y := y \times 2]\!]_0(w_2, w_3) \\
= \ & E(x := x + 1, (w_2, w_3)) + E(y := y \times 2, (w_2, w_3))
\end{aligned}$$

Again,

2: In this case choice is exactly nondeterministic choice: either one of $x := x + 1$ or $y := y \times 2$ will be executed.

G: This interpretation yields the maximum certainty degree of executing the composition, e.g. $E(x := x + 1, (w_2, w_3)) + E(y := y \times 2, (w_2, w_3)) = max\{0.7, 0.9\} = 0.9$.

R: In this action lattice, operator + picks the minimum value. This corresponds to choose the path that consumes less energy, e.g. $E(x := x + 1, (w_2, w_3)) + E(y := y \times 2, (w_2, w_3)) = min\{0.7, 0.9\} = 0.7$.

Note that nothing prevents the state space W from being infinite, because of completeness enforced upon **A**. However, one may only compute explicitly a truth value associated with a program execution when W is finite.

The second element to care about when computing the semantics is the interpretation of tests. Our goal is to introduce a notion of a test in an arbitrary dynamic logic generated by a parameter $\mathbf{A}$. As mentioned above, tests are written as $\varphi?$, for $\varphi \in \mathrm{Fm}^{\Gamma(\mathbf{A})}(\Delta)$. Their semantics resort, therefore, to the satisfaction relation for $\mathrm{Fm}^{\Gamma(\mathbf{A})}(\Delta)$, which is defined as follows:

Definition 9. *Given a complete action lattice* $\mathbf{A}$ *over a carrier* A*, the graded satisfaction relation for a model* $M \in \mathrm{Mod}^{\Gamma(\mathbf{A})}(\Delta)$*, consists of a function*

$$\models_{\Gamma(\mathbf{A})} : W \times \mathrm{Fm}^{\Gamma(\mathbf{A})}(\Delta) \to A$$

recursively defined by

- $(w \models_{\Gamma(\mathbf{A})} \top) = \top$
- $(w \models_{\Gamma(\mathbf{A})} \bot) = \bot$
- $(w \models_{\Gamma(\mathbf{A})} p) = [\![p]\!]_w$*, for any* $p \in T_\Sigma^P(X)$
- $(w \models_{\Gamma(\mathbf{A})} \varphi \wedge \varphi') = (w \models_{\Gamma(\mathbf{A})} \varphi) \cdot (w \models_{\Gamma(\mathbf{A})} \varphi')$
- $(w \models_{\Gamma(\mathbf{A})} \varphi \vee \varphi') = (w \models_{\Gamma(\mathbf{A})} \varphi) + (w \models_{\Gamma(\mathbf{A})} \varphi')$
- $(w \models_{\Gamma(\mathbf{A})} \varphi \to \varphi') = (w \models_{\Gamma(\mathbf{A})} \varphi) \to (w \models_{\Gamma(\mathbf{A})} \varphi')$
- $(w \models_{\Gamma(\mathbf{A})} \langle\pi\rangle\varphi) = \sum_{w' \in W} \big([\![\pi]\!](w, w'); (w' \models_{\Gamma(\mathbf{A})} \varphi)\big)$
- $(w \models_{\Gamma(\mathbf{A})} [\pi]\varphi) = \bigwedge_{w' \in W} \big([\![\pi]\!](w, w') \to (w' \models_{\Gamma(\mathbf{A})} \varphi)\big)$

The interpretation of tests in the classical, Boolean case is given by coreflexive relations $R_{\varphi?} = \{(w, w) | w \models \varphi\}$. In the generic setting of the present work this generalises to

$$[\![\varphi?]\!](w, w') = \begin{cases} (w \models_{\Gamma(\mathbf{A})} \varphi) & \text{if } w = w' \\ \bot & \text{otherwise} \end{cases}$$

Let us revisit Example 4 to interpret the conditional statement

$$\textbf{if } x \leq 3 \textbf{ then } x := x + 1 \textbf{ else } y := y \times 2$$

translated to $((x \leq 3?); x := x + 1) + ((((x \leq 3) \to \bot)?); y := y \times 2)$. Using the value computed for predicate $x \leq 3$, this leads to

$$\begin{aligned}
&[\![((x \leq 3)?; x := x + 1) + (((x \leq 3) \to \bot)?; y := y \times 2)]\!](w_2, w_3) = \\
&= [\![(x \leq 3)?; x := x + 1]\!](w_2, w_3) + [\![((x \leq 3) \to \bot)?; y := y \times 2]\!](w_2, w_3) \\
&= [\![(x \leq 3)?]\!](w_2, w_2); [\![x := x + 1]\!]_0(w_2, w_3) + [\![((x \leq 3) \to \bot)?]\!](w_2, w_2); [\![y := y \times 2]\!]_0(w_2, w_3) \\
&= (w_2 \models x \leq 3); E(x := x + 1, (w_2, w_3)) + (w_2 \models (x \leq 3) \to 0); E(y := y \times 2, (w_2, w_3)) \\
&= (w_2 \models x \leq 3); E(x := x + 1, (w_2, w_3)) + ((w_2 \models x \leq 3) \to (w \models 0)); E(y := y \times 2, (w_2, w_3))
\end{aligned}$$

which can be, once again, instantiated for the three action lattices under consideration, yielding

2: $(T \wedge T) \vee ((\top \to \bot) \wedge \top) = \top$. This interpretation coincides, as expected, with the standard **if-then-else** statement. In this case, only program $x := x + 1$ is executed, since $y = y \times 2$ is guarded by the test $((x \leq 3) \to \bot)?$ which has the value $\bot$ at state w_2.

G: $max\{min\{0.3, 0.7\}, \min\{0.3 \to 0, 0.9\}\} = 0.3$, which expresses the weighted choice of executing $x := x + 1$.

R: $min\{3 + 7, 0 + 9\} = 9$. In this situation, contrary to what happens in the previous cases, the assignment $y := y \times 2$ is executed. The value 9 stands for the energy consumed by the machine when executing such an assignment.

4 Bisimulation

The characterisation of relations that identify states with equivalent behaviours is crucial to support a set of development practices, including reuse, refinement and minimization of programs and models. On the logic view, these relations usually enjoy a modal invariance property, i.e. they preserve the satisfaction of formulas. We introduce in this section a parametric notion of bisimulation, and we prove its modal invariant for any $\Gamma(\mathbf{A})$. The bisimulation generalises the notion recently introduced by the authors in [5] in the context of fuzzy modal logic.

Definition 10 (Π-Bisimulation). *Let $\Delta = (\Sigma, \Pi)$ be a signature, X a set of variables, and $M = (W, E)$ and $M' = (W', E')$ two $\Gamma(\mathbf{A})$-models, for any linear action lattice $\mathbf{A}$.*

A Π-bisimulation from M to M' is a non empty relation $B \subseteq W \times W'$ such that whenever $w\ B\ w'$, the following conditions hold:

(Atoms) *for any $x \in X$, $r \in \mathbb{R}$, $[\![x]\!]_w(r) = [\![x]\!]_{w'}(r)$ and, for any $p \in T^P_\Sigma(X)$, $[\![p]\!]_w = [\![p]\!]_{w'}$*

(Fzig) *for any $u \in W$ and $\pi \in \Pi$, $[\![\pi]\!]_0(w, u) \leq \sum_{u' \in B[\{u\}]} [\![\pi]\!]_0(w', u')$*

(Fzag) *for any $u' \in W'$ and $\pi \in \Pi$, $[\![\pi]\!]_0(w', u') \leq \sum_{u \in B^{-1}[\{u'\}]} [\![\pi]\!]_0(w, u)$*

We write $w \sim w'$ whenever, there is a bisimulation B such that $(w, w') \in B$.

Next result establishes the well-known *word bisimulation result* on this generic graded settings. This result reduces the invariance property of formulas involving composed programs in $\mathrm{Prg}(\Sigma, X)$ to the one involving just the set of atomic programs Π. In other words, it reduces the modal invariance problem of a generated dynamic logic to the modal invariance of the underlying multi-valued logic.

Proposition 1. *Let $\mathbf{A}$ be a linear action lattice and (Σ, X) a data signature. Then, any Π-bisimulation over $\Gamma(\mathbf{A})$-models is a $\mathrm{Prg}(\Sigma, X)$-bisimulation.*

Proof. The proof is done by induction over the programs structure. Let $B \subseteq W \times W'$ be a bisimulation and $w \in W, w' \in W'$ such that $(w, w') \in B$.

The result for atomic programs is given by hypothesis. Let us prove the **(Fzig)** condition for programs $\pi; \pi'$. By induction hypothesis, let us assume that **(Fzig)** of B for π and π'. Hence, for any $v \in W$

$$[\![\pi]\!](w, v) \leq \sum_{v' \in B(v)} [\![\pi]\!](w', v') \tag{23}$$

holds. By (20) we have also that, for any $v \in W$ there is a $v'_v \in B(v)$ such that $\sum_{v' \in B(v)} [\![\pi]\!](w', v') = [\![\pi]\!](w', v'_v)$. Moreover, since $(v, v'_v) \in B$, we have by **(Fzig)** of B for π' that

$$[\![\pi']\!](v, u) \leq \sum_{u' \in B(u)} [\![\pi']\!](v'_v, u') \tag{24}$$

By (21) in (23) we get, for any $v \in W$,

$$[\![\pi]\!](w, v); [\![\pi']\!](v, u) \leq [\![\pi]\!](w', v'_v); \sum_{u' \in B(u)} [\![\pi']\!](v'_v, u') \tag{25}$$

and by (22),

$$\sum_{v \in W} [\![\pi]\!](w, v); [\![\pi']\!](v, u) \leq \sum_{v'_v \in W'} [\![\pi]\!](w', v'_v); \sum_{u' \in B(u)} [\![\pi']\!](v'_v, u') \tag{26}$$

Moreover, since $\{v'_v : v \in W\} \subseteq \{v' : v' \in W'\}$, and by (8), (2) and (3), we have that

$$\sum_{v'_v \in W'} [\![\pi]\!](w', v'_v); \sum_{u' \in B(u)} [\![\pi']\!](v'_v, u') \leq \sum_{u' \in B(u)} (\sum_{v' \in W'} ([\![\pi]\!](w', v'); [\![\pi']\!](v', u')) \tag{27}$$

By (26) and (27), we achieve $[\![\pi; \pi']\!](w, u) \leq \sum_{u' \in B(u)} [\![\pi; \pi']\!](w', u')$. The prove of **(Fzag)** condition is analogous.

For programs $\pi + \pi'$, we observe that

$$\begin{array}{ll|ll} & [\![\pi + \pi']\!](w, u) & & \sum_{u' \in B(u)} [\![\pi]\!](w', u') + \sum_{u' \in B(u)} [\![\pi']\!](w', u') \\ = & \{ \text{ interpretation of programs} \} & = & \{ \text{ definition of } + \} \\ & [\![\pi]\!](w, u) + [\![\pi']\!](w, u) & & \sum_{u' \in B(u)} [\![\pi + \pi']\!](w', u') \\ \leq & \{ \textbf{(Fzig)} \text{ and (22)} \} & & \end{array}$$

Finally, for programs π^* we observe that by definition of *

$$[\![\pi^*]\!](w, u) = \sum_{k \geq 0} [\![\pi]\!]^k(w, u) = [\![\pi]\!]^0(w, u) + [\![\pi]\!](w, u) + [\![\pi]\!]^2(w, u) + \ldots$$

But for each k, $[\![\pi]\!]^k(w, u) \leq \sum_{u \in B(u)} [\![\pi]\!]^k(w', u')$ by Fzig.
Hence,

$$\begin{array}{ll|ll} & \sum_{k \geq 0} [\![\pi]\!]^k(w, u) & & \sum_{u' \in B(u)} \left(\sum_{k \geq 0} [\![\pi]\!]^k(w', u') \right) \\ \leq & \{ \text{ (22)} \} & = & \{ \text{ definition of } * \} \\ & \sum_{k \geq 0} \left(\sum_{u' \in B(u)} [\![\pi]\!]^k(w', u') \right) & & \sum_{u' \in B(u)} [\![\pi^*]\!](w', u') \\ = & \{ \text{ (2) and (3)} \} & & \end{array}$$

■

Now we are in conditions to prove the modal invariance for $\Gamma(\mathbf{A})$ with $\mathbf{A}$ linear.

Theorem 1 (Modal invariance). *Let $\Delta = (\Sigma, X)$ be a signature, $\mathbf{A}$ a linear action lattice, and $M = (W, E)$ and $M' = (W', E')$ two $\Gamma(\mathbf{A})$-models for Δ. Then, for any $w \in W$, $w' \in W'$ such that $w \sim w'$ and for all formulas $\varphi \in \mathrm{Fm}^{\Gamma(\mathbf{A})}(\Delta)$,*

$$(M, w \models \varphi) = (M', w' \models \varphi)$$

Proof. We prove this result by induction on the structure of formulas.
For the invariance of the formula $\top$, note that $(M, w \models \top) = \top = (M', w' \models \top)$ and similarly for the formula $\bot$.
Invariance of $p \in T^P_\Sigma(X)$ is a direct consequence of **(Atoms)**,

$$(M, w \models p) = [\![p]\!]_w = [\![p]\!]_{w'} = (M', w' \models p).$$

For the invariance of formulas $\varphi \wedge \psi$, we observe that

$$(M, w \models \varphi \wedge \psi) = (M, w \models \varphi) \cdot (M, w \models \psi) =^{I.H.}$$

$$(M', w' \models \varphi) \cdot (M', w' \models \psi) = (M', w' \models \varphi \wedge \psi)$$

and the proof for the invariance of formulas $\varphi \vee \psi$ and $\varphi \to \psi$ can be proved similarly.

Now it just remains to prove sentences $\langle \pi \rangle \varphi$ and $[\pi]\varphi$. Since $\mathbf{A}$ is linear, we have by Proposition 1 that, it is enough to prove the invariance for formulas involving atomic programs $\pi_0 \in \mathrm{Prg}_0(\Sigma, X)$. For the invariance of formulas $\langle \pi_0 \rangle \varphi$, we observe that By **(Fzig)** condition we have

$$\forall u \in W, [\![\pi_0]\!]_0(w, u) \leq \sum_{u' \in\ E[\{u\}]} [\![\pi_0]\!]_0(w', u') = [\![\pi_0]\!]_0(w', u'_u) \text{ for some } u'_u \in W' \tag{28}$$

Since for every $u \in W, u'_u \in E[\{u\}]$, we have $u\ E\ u'_u$. By I. H., we have $(M, u \models \varphi) = (M', u'_u \models \varphi)$ and, by (28),

$$\forall u \in W, [\![\pi_0]\!]_0(w, u) \cdot (M, u \models \varphi) \leq [\![\pi_0]\!]_0(w', u'_u) \cdot (M, u'_u \models \varphi) \tag{29}$$

and, in particular,

$$\sum_{u \in W} ([\![\pi_0]\!]_0(w, u) \cdot (M, u \models \varphi)) \leq \sum_{u'_u : u \in W} ([\![\pi_0]\!]_0(w', u'_u) \cdot (M, u'_u \models \varphi)) \tag{30}$$

Since $\{u'_u : u \in W\} \subseteq \{u' : u' \in W'\}$ we have $\sum\{u'_u : u \in W\} \leq \sum\{u' : u' \in W'\}$ and by 30

$$\sum_{u \in W} ([\![\pi_0]\!]_0(w, u) \cdot (M, u \models \varphi)) \leq \sum_{u' \in W'} ([\![\pi_0]\!]_0(w', u') \cdot (M, u' \models \varphi)) \tag{31}$$

i.e.$(M, w \models \langle \pi_0 \rangle \varphi) \leq (M', w' \models \langle \pi_0 \rangle \varphi)$. Similarly we can prove $(M, w \models \langle \pi_0 \rangle \varphi) \geq (M', w' \models \langle \pi_0 \rangle \varphi)$ by using **(Fzag)** condition.

For the invariance of formulas $[\pi_0]\varphi$, with $\pi_0 \in \Pi$, since $w\ E\ w'$ we have by **(Fzig)**

$$\forall u \in W, [\![\pi_0]\!]_0(w,u) \leq \sum_{u' \in\ E[\{u\}]} [\![\pi_0]\!]_0(w',u') = [\![\pi_0]\!]_0(w',u'_u) \text{ for some } u'_u \in W' \tag{32}$$

Since for every $u \in W, u'_u \in E[\{u\}]$, we have $u \in W, u\ E\ u'_u$. Hence, by I.H.

$$(M,u \models \varphi) = (M',u'_u \models \varphi) \tag{33}$$

It follows from the definition of I that $x_0 \leq x_1$ implies $I(x_0,y) \geq I(x_1,y)$. Then, from (32) and (33) we have

$$\forall u \in W, I\big([\![\pi_0]\!]_0(w,u),(M,u \models \varphi)\big) \geq I\big([\![\pi_0]\!]_0(w',u'_u),(M',u'_u \models \varphi)\big)$$

and, in particular

$$\prod_{u \in W} (I\big([\![\pi_0]\!]_0(w,u),(M,u \models \varphi)\big)) \geq \prod_{u'_u : u \in W} (I\big([\![\pi_0]\!]_0(w',u'_u),(M',u'_u \models \varphi)\big)) \tag{34}$$

Since $\{u'_u : u \in W\} \subseteq \{u' : u' \in W'\}$, we have $\prod\{u'_u : u \in W\} \geq \prod\{u' : u' \in W'\}$ and hence

$$\prod_{u \in W} (I\big([\![\pi_0]\!]_0(w,u),(M,u \models \varphi)\big)) \geq \prod_{u' \in W'} (I\big([\![\pi_0]\!]_0(w',u'),(M',u' \models \varphi)\big)) \tag{35}$$

Therefore $(M,w \models [\pi_0]\varphi) \geq (M',w' \models [\pi_0]\varphi)$. The proof for $(M,w \models [\pi_0]\varphi) \leq (M',w' \models [\pi_0]\varphi)$ is analogous. ■

We now provide an illustration for the introduced notion of bisimulation.

Example 5. Consider the $\Gamma(\mathbf{G})$-models $M = (W,V,E)$, with $W = \{w_1,w_2,w_3,w_4\}$ and $M' = (W',V',E')$, with $W' = \{w'_1,w'_2,w'_3,w'_4\}$, and the programs $\Pi = \{x := x{+}1, x := 3\}$, with $E(x := x{+}1,(w_1,w_2)) = 0.9$, $E(x := 3,(w_1,w_3)) = 0.8$, $E(x := 3,(w_1,w_4)) = 0.7$, $E(x := x+1,(w'_1,w'_2)) = 0.9$, $E(x := 3,(w'_1,w'_3)) = 0.8$, $E(x := x+1,(w'_1,w'_4)) = 0.6$.

To show that the relation $B = \{(w_1,w'_1),(w_2,w'_2),(w_2,w'_4),(w_3,w'_3),(w_4,w'_3)\}$ is a bisimulation from M to M', the **(Fzig)** and **(Fzag)** conditions of Definition 10 need to be satisfied. To exemplify, only the calculations for the case $w_1 \sim w'_1$ are provided, since the other pairs can be verified analogously.

(Fzig):

$[\![x := x+1]\!]_0(w_1,w_2) \leq max\{[\![x := x+1]\!]_0(w'_1,w'_2),[\![x := x+1]\!]_0(w'_1,w'_4)\}$
$\Leftrightarrow 0.9 \leq max\{0.9,0.6\} \Leftrightarrow 0.9 \leq 0.9$
$[\![x := 3]\!]_0(w_1,w_3) \leq [\![x := 3]\!]_0(w'_1,w'_3) \Leftrightarrow 0.8 \leq 0.8$
$[\![x := 3]\!]_0(w_1,w_4) \leq [\![x := 3]\!]_0(w'_1,w'_3) \Leftrightarrow 0.7 \leq 0.8$

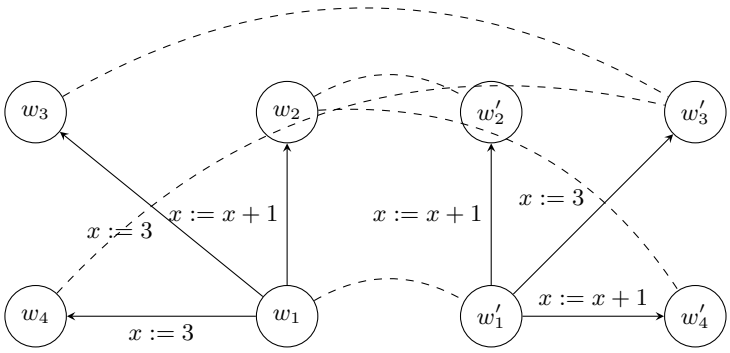


Fig. 2. Two bisimilar $\Gamma(\mathbf{G}) - models$

(Fzag):

$$\begin{aligned}
&[\![x := x + 1]\!]_0(w_1', w_2') \leq [\![x := x + 1]\!]_0(w_1, w_2) = 0.9\\
&[\![x := x + 1]\!]_0(w_1', w_4') \leq [\![x := x + 1]\!]_0(w_1, w_2) = 0.9\\
&[\![x := 3]\!]_0(w_1', w_3') \leq \max\{[\![x := 3]\!]_0(w_1, w_3), [\![x := 3]\!]_0(w_1, w_4)\\
\Leftrightarrow &0.8 \leq max\{0.8, 0.7\} \Leftrightarrow 0.8 \leq 0.8
\end{aligned}$$

5 Conclusions and future work

This paper extended the process of systematic generation of multi-valued dynamic logics from the original propositional case [9], to 'fully-fledged' programs, which incorporate variables and assignments. As before, the method is parametric on an action lattice which supports both a computational model in which programs are defined, and a truth space, suitable to handle different aspects of the application domain. Both states, specified by assignments of real values to variables, and transitions between them have an associated 'weight', i.e. a value taken from the carrier of a action lattice. As detailed in the examples discussed, the notion of 'weight' as formalised in an action lattice, is the real parameter of this process. Actually, they can capture quite a range of effects: from the degree of vagueness of an execution, to the cost of resources. The notion of bisimulation presented in Section 4 generalises previous work done by the authors [5], in the sense that a generic action lattice is considered as a parameter of the generated logics. A prominent application of dynamic logic lies in the field of formal verification of programs, as a simplification of the deductive apparatus of Hoare logic. In such formalism, the correctness of a program is proved by stating the validity of an Hoare triple $\varphi\{\pi\}\psi$. As it is well known, the validity of the dynamic logic formula $w \models \varphi \rightarrow [\pi]\psi$, is an abstraction of such proof. In this sense, the multi-valued nature of the logics generated in this paper may present a proper formalism to state program correctness in a multi-valued setting as well:

the "degree of correctness" of a program may be computed as the value, in the parameter **A**, of the above dynamic logic formula. Motivated by this example, it is our intention to include a calculi for such logics as part of our research agenda.

References

1. A. Baltag and S. Smets. The dynamic turn in quantum logic. *Synthese*, 186(3):753–773, 2012. doi:10.1007/s11229-011-9915-7.
2. M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979. doi:10.1016/0022-0000(79)90046-1.
3. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic logic*. MIT Press, Cambridge, MA, USA, 2000.
4. R. Hennicker, A. Madeira, and A. Knapp. A hybrid dynamic logic for event/data-based systems. In Reiner Hähnle and Wil M. P. van der Aalst, editors, *Fundamental Approaches to Software Engineering - 22nd Int. Conf., FASE 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11424 of *Lecture Notes in Computer Science*, pages 79–97. Springer, 2019. doi:10.1007/978-3-030-16722-6_5.
5. M. Jain, A. Madeira, and M. A. Martins. A fuzzy modal logic for fuzzy transition systems. *Electr. Notes Theor. Comput. Sci.*, (in print).
6. A. Knapp, T. Mossakowski, M. Roggenbach, and M. Glauer. An institution for simple UML state machines. In Alexander Egyed and Ina Schaefer, editors, *Fundamental Approaches to Software Engineering - 18th Int. Conf., FASE 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9033 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015. doi:10.1007/978-3-662-46675-9_1.
7. D. Kozen. A probabilistic PDL. *J. Comput. Syst. Sci.*, 30(2):162–178, 1985. doi:10.1016/0022-0000(85)90012-1.
8. D. Kozen. *The design and analysis of algorithms*. Springer-Verlag New York, 1992.
9. A. Madeira, R. Neves, and M. A. Martins. An exercise on the generation of many-valued dynamic logics. *Journal of Logical and Algebraic Methods in Programming*, 1:1–29, 2016. doi:10.1016/j.jlamp.2016.03.004.
10. D. Peleg. Concurrent dynamic logic. *J. ACM*, 34(2):450–479, 1987. doi:10.1145/23005.23008.
11. A. Platzer. *Logical Analysis of Hybrid Systems - Proving Theorems for Complex Dynamics*. Springer, 2010. doi:10.1007/978-3-642-14509-4.